

MULTIPLE TEMPLATE LEARNING FOR STRUCTURED PREDICTION

QI MAO AND IVOR W. TSANG

ABSTRACT. Conditional random field (CRF) and Structural Support Vector Machine (SVM) are two state-of-the-art algorithms for structured prediction, which captures the interdependency among output variables. The success of these algorithms is attributed to the fact that their discriminative models can account for overlapping features on the whole input observations. These features are usually generated by applying a given set of templates on labeled data, but improper templates may lead to degraded performance. To alleviate this issue, in this paper, we propose a novel multiple template learning paradigm to learn structured prediction and the importance of each template simultaneously, so that arbitrary templates could be added into the learning model without caution. This paradigm can be formulated as a special multiple kernel learning problem with exponential number of constraints. Then we introduce an efficient cutting plane algorithm to solve this problem in the primal. We also evaluate the proposed learning paradigm on two widely-studied structured prediction tasks, *i.e.* sequence labeling and dependency parsing. Extensive experimental results show that the proposed method outperforms CRFs and Structural SVMs due to exploiting the importance of each template.

1. INTRODUCTION

Structured prediction [13, 28, 30] has been successfully applied to the problems with strong interdependence among the output variables. In the realm of Natural Language Processing (NLP), many tasks could be formulated as structured prediction problems. A typical example is part-of-speech tagging which assigns a specific part-of-speech tag to each token of an input sentence. The tag of one token is strongly correlated with the tags of its neighbors under the linear chain dependency [13, 30]. More complicated structured output dependencies could be a tree or graphs, such as Context-Free Grammar (CFG) [30], dependency parsing tree [17, 18], and factor graph for relation extraction [37]. Note that there exist exact inference methods for sequence and tree structures. For the tasks with general output structures (*e.g.*, the pairwise fully connected undirected graph), however, the exact inference problem is intractable. In such cases, approximate inference is usually pursued to obtain an approximate solution [11].

The major advantage of structured prediction models such as Conditional Random Fields (CRFs) and Structural Support Vector Machines (SVMs) is that their learning models can easily integrate prior knowledge of a specific domain by feature engineering. For example, the discriminative models of CRFs can account for overlapping features (*e.g.*, first-order or even higher-order linear chain) on the whole observation sequence [13]. On the other hand, Structural SVMs [30, 12] relies on the

Qi Mao and Ivor W. Tsang are with School of Computer Engineering, Nanyang Technological University, Singapore 639798, e-mail {QMAO1,IvorTsang}@ntu.edu.sg.

joint feature maps over the input-output pairs, where features can be represented equivalently as that of CRFs.

During last decade, structured prediction algorithms take more effort on how to model the interdependence among the output variables, but less consideration is taken on the feature engineering which is a non-trivial task for general users. We observe that different kinds of rules are used to extract features from input sentences in the sequence learning and dependency parsing. The arbitrary non-independent features are usually extracted from a given input-output pair using a set of predefined templates (rules or feature functions). Templates can be arbitrarily defined according to specific applications by exploring any internal or external knowledge as much as possible, and then are added into the learning models in order to boost prediction performance. However, features generated from arbitrary templates may be redundant or non-informative. Structured prediction models, such as CRFs and Structural SVMs, only treat the features generated from each template equally without exploiting the importance of each template or its generated features. Therefore, some improper templates may generate conflicting or noisy features which can degrade these structured prediction models.

In this paper, we propose a Multiple Template Learning (MTL) paradigm to learn the weight of each template and the structured prediction model, simultaneously. Specifically, given a set of predefined templates, the features extracted from each individual template can be inherently formed as a group. Learning the weighting of these groups is formulated as a Multiple Kernel Learning (MKL) problem. This specific MKL problem usually involves exponential number of constraints due to the interdependence among output variables. We propose to solve this MKL problem in the primal by an efficient cutting plane algorithm. The proposed MTL paradigm can be easily instantiated for specific applications which inherits from Structural SVMs. Moreover, two well-known structured prediction tasks, sequence labeling and dependency parsing, are showcased in this paper. Extensive experimental results demonstrate that the proposed paradigm can automatically learn the importance of each templates for the structured prediction tasks. The learned weights can avoid degraded performance which is caused by adding poorly designed or even conflicting templates into the learning model, so users can define templates without cautions. Moreover, MTL helps boost the prediction performance by weighting the features among different groups.

The rest of this paper is organized as follows: Section 2 briefly reviews related work of structured prediction and multiple kernel learning. In Section 3, we first describe the relationship between features and templates in natural language processing tasks. Then the proposed MTL framework for structured prediction is presented. Section 4 and Section 5 discuss two showcases, sequence labeling and dependency parsing, in the MTL framework and related experimental results, respectively. Finally, Section 6 gives conclusive remarks.

2. RELATED WORK

There are a large number of real applications which have already been explored by structured prediction. We mainly review the work closely related to this paper such as sequence learning and dependency parsing, where inference problems can be solved exactly. The primary idea of sequence labeling is to learn from observation

sequences and then predict label sequences instead of individual class labels. Fundamental tasks in NLP, such as Chinese word segmentation [23, 38], part-of-speech tagging [13, 1, 12], and named entity recognition [16, 22, 10], can achieve great improvements due to the development of label sequence learning models, such as Hidden Markov models (HMMs) [24], Conditional Random Fields (CRFs) [13], Hidden Markov Support Vector Machines (HM-SVM)[1] and Structural SVMs [30, 12]. In addition, sequence labeling has been successfully applied in other areas such as speech recognition, computational biology and system identification. Different from sequence labeling, dependency parsing is to learn a directed tree structure which captures the syntactic relationships between two words in a sentence. Graph-based method [8, 17, 19] and transition-based method[36, 20, 3] are considered to be the state-of-the-art for dependency parsing[21]. It has been used ranging from question answering [33] to relation extraction [7].

The proposed MTL extends Structural SVM [30, 12] for structured and interdependent output variables to take care of the feature engineering problem. Structural SVM takes the joint feature map $\Psi(\mathbf{x}, \mathbf{y})$ over the input-output pair (\mathbf{x}, \mathbf{y}) with the input $\mathbf{x} \in \mathcal{X}$ and its output $\mathbf{y} \in \mathcal{Y}$ by concatenating different kinds of features together. The discriminant function is defined as $f(\mathbf{x}) = \arg \max_{\mathbf{y}' \in \mathcal{Y}} h(\mathbf{x}, \mathbf{y}')$ where compatibility functions $h(\mathbf{x}, \mathbf{y})$ assume to be linear in $\Psi(\mathbf{x}, \mathbf{y})$, i.e. $h(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$ as a \mathbf{w} -parameterized function. Given a convex loss function $L(\mathbf{x}, \mathbf{y}; \mathbf{w})$, Structural SVM is formulated as a minimization of the regularized empirical risk:

$$(1) \quad \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^n L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}),$$

where the common choices are the structured hinge loss and logistic loss in CRFs:

$$(2) \quad L_{hinge}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \max_{\mathbf{y}' \in \mathcal{Y}} \Delta(\mathbf{y}, \mathbf{y}') + h(\mathbf{x}, \mathbf{y}'; \mathbf{w}) - h(\mathbf{x}, \mathbf{y}; \mathbf{w})$$

$$(3) \quad L_{CRFs}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \log \sum_{\mathbf{y}' \in \mathcal{Y}} \exp(h(\mathbf{x}, \mathbf{y}'; \mathbf{w}) - h(\mathbf{x}, \mathbf{y}; \mathbf{w}))$$

where $\Delta(\mathbf{y}, \mathbf{y}')$ is a user-defined cost function. This cost penalizes the output \mathbf{y}' violating a margin constraint involving $\mathbf{y}' \neq \mathbf{y}$.

Feature selection for CRFs has been explored for activity recognition by ℓ_1 regularization [31]. Progressively reducing the features by ℓ_1 regularization method may degrade the performance greatly due to the interdependence among features in the same group. MTL considers that features are naturally group together by templates in terms of Multiple Kernel Learning which was first proposed by [14], and the connection with $\ell_{2,1}$ mixed norm was proposed by [4] to learn group sparsity for independent output variables. There is lots of work focusing on convergence and scalability [27, 25, 34, 29]. [39] formulated MKL for multiple classification problem in terms of structural prediction, but only focused on multiple classification. It is intractable for these MKL methods to deal with structured prediction due to the exponential number of constraints. Recently, [15] proposed multiple kernel learning for structural prediction training in online mode; while MTL is trained in batch mode. Moreover, we propose to use templates to generate natural feature groups for MKL.

3. MULTIPLE TEMPLATE LEARNING FOR STRUCTURED PREDICTION

Conditional Random Fields (CRFs) and Structural SVMs can be used to model many structured prediction problems, such as sequence labeling with first-order or even higher-order linear chain, and syntactic parsing with directed tree structure. More general structures can be possible if a feasible inference method exists. CRFs are graphical models by defining conditional probability density function over the given graph structure, while Structural SVMs incorporate the structural information by defining joint feature mapping under large margin theory. Even though they model the same problem under different principles, they all confront the same difficulties: how to extract features from structured inputs for the learning algorithms so as to obtain better performance. There are lots of tasks where features are extracted by applying a set of predefined templates. Our proposed Multiple Template Learning (MTL) framework mainly focuses on how to efficiently and effectively manage the given set of templates and learn a better structured prediction model simultaneously.

3.1. Feature v.s. Template. For the problems with structured inputs, features are usually not explicitly defined. For instance, part-of-speech tagging in the area of NLP is to label each token with a specific tag in the given token sequence (sentence). In addition to the current token, the neighbors and their associated tags can be considered as the important features to determine the tag of the current token. All these intuitive information can be represented by feature functions called templates. In this paper, we consider the templates as some predefined rules which are used for feature extraction task.

Given a training dataset $\mathcal{D} = \{(\mathcal{X}_i, \mathcal{Y}_i)\}_{i=1}^n$ with n structured input-output pairs $(\mathcal{X}_i, \mathcal{Y}_i)$ where $\mathcal{X}_i \in \mathbb{X}$ and $\mathcal{Y}_i \in \mathbb{Y}$ can be any structures, for example, sequence, tree, or general graph. Assume that there are m templates denoted by the functional $\kappa_j(\cdot), \forall j = 1, \dots, m$ over the domain of structural input-output pairs. By applying $\kappa_j(\cdot)$ to all pairs of \mathcal{D} , we can instantiate a set of features $\kappa_j(\mathcal{D}) = \{\kappa_j^1, \dots, \kappa_j^{d_j}\}$ with d_j different features. Therefore, the set of features over \mathcal{D} are $\kappa(\mathcal{D}) = \{\kappa_1(\mathcal{D}), \dots, \kappa_m(\mathcal{D})\}$. The size of total features $|\kappa(\mathcal{D})|$ may be smaller than $\sum_{j=1}^m |\kappa_j(\mathcal{D})|$ since one feature may be generated from more than one template. Each input-output pair $(\mathcal{X}_i, \mathcal{Y}_i)$ now can be represented by a real value feature vector $\Phi(\mathcal{X}_i, \mathcal{Y}_i)$ using the same set of features in $\kappa(\mathcal{D})$. Since $\kappa(\mathcal{D})$ is extracted from \mathcal{D} and a small subset of features in $\Phi(\mathcal{X}_i, \mathcal{Y}_i)$ could be activated, so each instance may have a very sparse feature representation.

The previous methods for structured inputs and outputs did not consider the properties of features, and directly used $\kappa(\mathcal{D})$ as the feature representation. Taking part-of-speech tagging for example, CRFs use the concatenation of all the different features instantiated from each template and the real-value of each feature is the occurred frequency. Actually, the subset of features $\kappa_j(\mathcal{D})$ applied by the j th template can be naturally formulated as one group. Each group of features stands for some specific meaning of the applied template, such as word, bi-gram, the distance, direction, and position related to the current token. Moreover, templates can be either defined arbitrarily by persons without any prior knowledge, or designed specially by domain experts. Due to the diversity of these templates, the importance of each templates should be different.

It is desirable to have a principle way to interpret these templates. In particular, a proper template weighting can prune away poorly designed or conflicting templates, and amplify the effective templates, thereafter templates can be designed without cautions. To this end, in the next subsection, we propose a novel structured prediction model with group sparsity, where features generated from the same template naturally form a group, so as to interpret the importance of templates. The proposed model can be essentially deemed as a special Multiple Kernel Learning (MKL) problem, where the base kernels are defined in accord with templates. Then we solve this MKL in the primal by an efficient cutting plane algorithm.

3.2. Model Formulation. Given a training dataset \mathcal{D} , each input-output pair $(\mathcal{X}_i, \mathcal{Y}_i)$ can be represented by $\Phi(\mathcal{X}_i, \mathcal{Y}_i) = [\Phi_1(\mathcal{X}_i, \mathcal{Y}_i); \dots; \Phi_m(\mathcal{X}_i, \mathcal{Y}_i)]$ using the group representation of $\kappa(\mathcal{D})$ where semicolon is used to concatenate column vectors, for concise representation. The goal of structure learning is to learn hypotheses

$$(4) \quad f : \mathbb{X} \rightarrow \mathbb{Y}.$$

According to Structural SVMs, the compatibility function $F : \mathbb{X} \times \mathbb{Y} \rightarrow \mathbb{R}$ over the input-output pairs are pursued and the prediction function can be derived by maximizing F over the output space \mathbb{Y} for a given input $\mathcal{X} \in \mathbb{X}$. The general hypotheses f are the parameterized functions with parameter vector \mathbf{w} as

$$(5) \quad f(\mathcal{X}; \mathbf{w}) = \arg \max_{\mathcal{Y} \in \mathbb{Y}} F(\mathcal{X}, \mathcal{Y}; \mathbf{w}).$$

The linear parametric representation of F is usually used as $F(\mathcal{X}, \mathcal{Y}; \mathbf{w}) = \mathbf{w}^T \Phi(\mathcal{X}, \mathcal{Y})$. For structural outputs, the standard zero-one cost function frequently used in classification is not appropriate. Most applications need specific cost function, so we define the general cost function $\Delta(\mathcal{Y}, \mathcal{Y}')$ to quantify the cost if the instance with true output \mathcal{Y} is assigned to be \mathcal{Y}' . The margin re-scaling with general loss functions and linear penalty term can be formulated as a minimization of the regularized empirical risk :

$$(6) \quad \begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \forall i, \forall \mathcal{Y}'_i \in \mathbb{Y} : \mathbf{w}^T \delta \Phi^i(\mathcal{Y}'_i) \geq \Delta(\mathcal{Y}_i, \mathcal{Y}'_i) - \xi_i \end{aligned}$$

where C is a trade-off parameter between training error minimization and margin maximization, and $\delta \Phi^i(\mathcal{Y}') = \Phi(\mathcal{X}_i, \mathcal{Y}_i) - \Phi(\mathcal{X}_i, \mathcal{Y}'_i)$.

As mentioned in Section 3.1, group sparsity could be applied to this problem in terms of the natural formed groups of features. According to Support Kernel Machines (SKMs) [4], Problem (6) can be readily formulated as SKMs with group feature representation $\Phi(\mathcal{X}_i, \mathcal{Y}_i) = [\Phi_1(\mathcal{X}_i, \mathcal{Y}_i); \dots; \Phi_m(\mathcal{X}_i, \mathcal{Y}_i)]$ as follows,

$$(7) \quad \begin{aligned} \min_{\mathbf{w}, \xi} \quad & \frac{1}{2} \left(\sum_{j=1}^m \|\mathbf{w}_j\| \right)^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \forall i, \forall \mathcal{Y}'_i \in \mathbb{Y} : \sum_{j=1}^m \mathbf{w}_j^T \delta \Phi_j^i(\mathcal{Y}'_i) \geq \Delta(\mathcal{Y}_i, \mathcal{Y}'_i) - \xi_i, \end{aligned}$$

where $\mathbf{w} = [\mathbf{w}_1; \dots; \mathbf{w}_m]$. The number of constraints in Problem (7) depends the specific structure in the space of \mathbb{Y} . In this paper, we mainly focus on sequence or tree structure which usually induces exponential number of constraints. Therefore,

the general SKMs, as well as the state-of-the-art MKLs, cannot be applied here. Recall that most algorithms focus on the dual problem of MKLs, but we propose an efficient cutting plane algorithm to solve Problem (7) in the primal form.

3.3. Multiple Kernel Learning Trained in the Primal. Problem (7) is an MKL problem with exponential number of constraints. The mixed norm regularizer makes it even hard to be solved. We first formulate it as 1-slack formulation,

$$(8) \quad \min_{\mathbf{w}, \xi} \quad \frac{1}{2} \left(\sum_{j=1}^m \|\mathbf{w}_j\| \right)^2 + C\xi$$

$$\text{s.t.} \quad \forall i, \forall \mathcal{Y}'_i \in \mathbb{Y} : \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \mathbf{w}_j^T \delta \Phi_j^i(\mathcal{Y}'_i) \geq \frac{1}{n} \sum_{i=1}^n \Delta(\mathcal{Y}_i, \mathcal{Y}'_i) - \xi.$$

And then, we propose Algorithm 1 to solve Problem (8). This algorithm constructs a working set \mathcal{W} iteratively. In each iteration, the most violated constraint is found, and added into working set \mathcal{W} . Then a subproblem is solved over \mathcal{W} in order to obtain a new solution \mathbf{w} . The algorithm stops if no constraint is found with the desired precision or the maximum number of iterations is reached.

Algorithm 1 Multiple Template Learning

- 1: **Input:** $\mathcal{D} = \{(\mathcal{X}_1, \mathcal{Y}_1), \dots, (\mathcal{X}_n, \mathcal{Y}_n)\}, C, \epsilon$
 - 2: $\mathbf{w} = \mathbf{0}, \mathcal{W} = \emptyset$,
 - 3: **repeat**
 - 4: Find the most violated $(\hat{\mathcal{Y}}_1, \dots, \hat{\mathcal{Y}}_n)$
 - 5: $\mathcal{W} := \mathcal{W} \cup \{(\hat{\mathcal{Y}}_1, \dots, \hat{\mathcal{Y}}_n)\}$
 - 6: Obtain \mathbf{w} by solving general QCQP subproblem over \mathcal{W}
 - 7: **until** ϵ -optimal
 - 8: **return** \mathbf{w}
-

Assume that finding the most violated $(\hat{\mathcal{Y}}_1, \dots, \hat{\mathcal{Y}}_n)$ can be achieved given the model parameter \mathbf{w} . After s iterations, we can construct a set of most violated label \mathcal{W}^s , the subproblem in Algorithm 1 is formulated as

$$(9) \quad \min_{\mathbf{w}, \xi \geq 0} \quad \frac{1}{2} \left(\sum_{j=1}^m \|\mathbf{w}_j\| \right)^2 + C\xi$$

$$\text{s.t.} \quad \xi \geq q^r + \sum_{j=1}^m \mathbf{w}_j^T \mathbf{p}_j^r, \forall r = 1, \dots, s$$

where $\mathbf{p}_j^r = -\frac{1}{n} \sum_{i=1}^n \delta \Phi_j^i(\hat{\mathcal{Y}}_i^r)$ and $q^r = \frac{1}{n} \sum_{i=1}^n \Delta(\mathcal{Y}_i, \hat{\mathcal{Y}}_i^r)$. Note that subproblem (9) has s constraints. The conic dual of (9) can be readily derived as

$$(10) \quad \max_{\alpha \in \mathcal{A}_s} \max_{\theta} \quad -\theta + \sum_{r=1}^s \alpha_r q^r$$

$$\text{s.t.} \quad \frac{1}{2} \alpha^T Q^j \alpha \leq \theta, \forall j = 1, \dots, m$$

where $\mathcal{A}_s = \{\sum_{r=1}^s \alpha_r \leq C, \alpha_r \geq 0, \forall r = 1, \dots, s\}$, and $Q_{r,r'}^j = \langle \mathbf{p}_j^r, \mathbf{p}_j^{r'} \rangle$. For completeness, we give the derivation in Appendix A. Problem (10) is in form of a

quadratically constrained quadratic programming (QCQP) problem, which is similar to the multiple kernel learning problem with small size of constraints. Furthermore, the primal solutions can be recovered by $\mathbf{w}_j = \mu_j \sum_{r=1}^s \alpha_r \mathbf{p}_j^r, \forall j = 1, \dots, m$ where μ is the Lagrangian multiplier of each corresponding constraint in (10).

Empirically, the number of iteration s needed for Algorithm 1 to reach ϵ -optimal convergence is very small. Therefore, the QCQP problem in (10) with $s+1$ variables and $m+1$ constraints can be solved efficiently by a QCQP toolbox, such as Mosek [2]. Since Mosek simultaneously solves the primal and its dual form, the weights μ for each group of features can be obtained at the same time. Alternatively, one can apply other efficient MKL algorithms to solve (10).

It is worth mentioning that the proposed method obtains $\mathbf{w}_j = \mu_j \sum_{r=1}^s \alpha_r \mathbf{p}_j^r, \forall j = 1, \dots, m$; while Structural SVMs do not consider the weights μ , which means a uniform weighting for all the group of features or templates.

The procedure for finding most violated constraints is to solve

$$(11) \quad (\hat{\mathcal{Y}}_1, \dots, \hat{\mathcal{Y}}_n) = \arg \max_{(\mathcal{Y}'_1, \dots, \mathcal{Y}'_n) \in \mathbb{Y}^n} \sum_{i=1}^n \Delta(\mathcal{Y}_i, \mathcal{Y}'_i) - \sum_{i=1}^n \sum_{j=1}^m \mathbf{w}_j^T \delta \Phi_j^i(\mathcal{Y}'_i),$$

where the definition of cost $\Delta(\mathcal{Y}_i, \mathcal{Y}'_i)$ and the feature mapping Φ depend on the specific tasks. The input-output pairs in \mathcal{D} are generally considered i.i.d., so Problem (11) can generally decomposed into n independent optimization problem as

$$(12) \quad \hat{\mathcal{Y}}_i = \arg \max_{\mathcal{Y}'_i \in \mathbb{Y}} \Delta(\mathcal{Y}_i, \mathcal{Y}'_i) - \sum_{j=1}^m \mathbf{w}_j^T \delta \Phi_j^i(\mathcal{Y}'_i), \forall i = 1, \dots, n.$$

The stopping criterion of Algorithm 1 is defined as $R_{emp}(\mathbf{w}_s) - R_s(\mathbf{w}_s) < \epsilon$ where the risk of upper bound and lower bound of Problem (8) are

$$R_{emp}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \max_{\mathcal{Y}'_i \in \mathbb{Y}} \left(\Delta(\mathcal{Y}_i, \mathcal{Y}'_i) - \sum_{j=1}^m \mathbf{w}_j^T \delta \Phi_j^i(\mathcal{Y}'_i) \right),$$

$$R_s(\mathbf{w}) = \max_{r=1, \dots, s} \left(\sum_{j=1}^m \mathbf{w}_j^T \mathbf{p}_j^r + q^r \right).$$

The maximum number of iterations could also be used to terminate the algorithm considering the time and space limitations.

In the next section, we tackle specific tasks, namely sequence labeling and dependency parsing, where we will introduce the specific procedure for finding the most violated constraints for the two instantiations in details. In addition, we can do prediction by solving an inference Problem (5) given an input $\mathcal{X} \in \mathbb{X}$ after the parameter \mathbf{w} is learned.

4. MULTIPLE TEMPLATE LEARNING FOR SEQUENCE LABELING

In this section, we investigate multiple template learning algorithm in sequence labeling problems. We restrict the output structure to the linear chain structure where the class label of the current node depends on its neighbors only. Next, we illustrate the problem of sequence labeling, and define the feature groups by templates, as well as the methods for finding most violated constraints. Extensive experiments are performed on three sequence labeling tasks: Chinese word segmentation, Chinese part-of-speech tagging, and language-independent named entity recognition.

4.1. Sequence Labeling. Sequence Labeling is to learn from an observation sequence $\mathbf{o} \in \mathbb{O}$ and then predict a label sequence $\mathbf{y} \in \mathbb{Y}$ instead of individual class labels. Formally, given a training dataset $\{(\mathbf{o}_i, \mathbf{y}_i)\}_{i=1}^n$ with n sequence pairs $(\mathbf{o}_i, \mathbf{y}_i)$ where an observation sequence $\mathbf{o}_i = (o_{i,1}, \dots, o_{i,l_i})$ and a class label sequence $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,l_i})$ have the same size l_i , the task of sequence labeling is to learn the hypotheses

$$(13) \quad g : \mathbb{O} \rightarrow \mathbb{Y},$$

where the observation \mathbf{o} and the label \mathbf{y} in the pair (\mathbf{o}, \mathbf{y}) have the same finite size l .

There are a large amount of applications which could be formulated as sequence labeling problems. In this paper, we explore the advantages of our proposed MTL in Section 3 on Chinese word segmentation, Chinese part-of-speech tagging and language-independent named entity recognition. Taking Chinese part-of-speech tagging for example, given a word sequence in Figure 1(b), the goal is to predict a label class sequence $\mathbf{y} = (ns \ v \ t \ n)$ where ns is location name, v is a verb, t is a time word and n is a noun word. However, learning algorithms cannot be directly applied into the string representation of a sequence. In the following sections, we introduce the general feature representation strategy for sequence labeling problem and the corresponding inference methods.

4.2. Feature Representation. As stated in Section 3.1, the set of features can be extracted by a set of predefined templates. For sequence labeling problem, the features are first extracted from an observation sequence, and then convolutes with class labels. We define the feature symbol $\kappa_j(t, \mathbf{o})$ which stands for the feature extracted from the t^{th} token in the observation sequence \mathbf{o} . By applying this template to all the observation sequences in the dataset \mathcal{D} , we can collect a set of features $\kappa_j(\mathcal{D}) = \{\kappa_j^1, \dots, \kappa_j^{d_j}\}$. The total set of features is $\kappa(\mathcal{D}) = \{\kappa_1(\mathcal{D}), \dots, \kappa_m(\mathcal{D})\}$. Now, the token in t^{th} position of the observation sequence \mathbf{o} can be represented by a feature vector $\phi_j(t, \mathbf{o}) = [\delta(\kappa_j(t, \mathbf{o}), \kappa_j^1); \dots; \delta(\kappa_j(t, \mathbf{o}), \kappa_j^{d_j})]$ where $\delta(a, b) = 1$, if $a = b$, otherwise 0. Real value function can also be used to replace indicator function $\delta(., .)$.

As shown in [30], by introducing direct tensor product operator \otimes and canonical (binary) representation of labels $y \in \mathcal{Y}$ where $\mathcal{Y} = \{y_1, \dots, y_k\}$ is a set of unique class labels with the size of k , feature functions can be represented by a zero-one column vector:

$$(14) \quad \Lambda(y) = [\delta(y_1, y); \delta(y_2, y); \dots; \delta(y_k, y)] \in \{0, 1\}^k,$$

with $\langle \Lambda(y), \Lambda(y') \rangle = \delta(y, y')$, and the direct tensor operator is defined as $\otimes : \mathbb{R}^\ell \times \mathbb{R}^k \rightarrow \mathbb{R}^{\ell \cdot k}$, $(\mathbf{a} \otimes \mathbf{b})_{i+(j-1)\ell} = a_i \cdot b_j$.

Based on the features defined on observation sequence and class label sequence, we can define the joint feature, such as state-observation feature

$$(15) \quad \Phi_j(t, \mathbf{o}, \mathbf{y}) = \phi_j(t, \mathbf{o}) \otimes \Lambda(y_t), \forall j = 1, \dots, m,$$

and state-state feature

$$(16) \quad \Psi(t, t-1, \mathbf{y}) = \Lambda(y_{t-1}) \otimes \Lambda(y_t).$$

The features over observation-label sequence pairs (\mathbf{o}, \mathbf{y}) can be defined as the summation of feature vectors over all tokens in an observation sequence as

$$(17) \quad \Phi_j(\mathbf{o}, \mathbf{y}) = \sum_{t=1}^l \phi_j(t, \mathbf{o}) \otimes \Lambda(y_t), \forall j = 1, \dots, m,$$

and

$$(18) \quad \Psi(\mathbf{y}) = \sum_{t=2}^l \Lambda(y_{t-1}) \otimes \Lambda(y_t).$$

Finally, we can obtain a feature vector $\Phi(\mathbf{o}, \mathbf{y}) = [\Phi_1(\mathbf{o}, \mathbf{y}); \dots; \Phi_m(\mathbf{o}, \mathbf{y}); \Psi(\mathbf{y})]$ to represent the observation-label sequence pairs (\mathbf{o}, \mathbf{y}) . In this paper, we use the indicator function to construct features since the value in the each entry of $\Phi(\mathbf{o}, \mathbf{y})$ is the frequency of this feature appeared in the given observation-label pair (\mathbf{o}, \mathbf{y}) . And also for simplicity, we only implement the state-state transition feature without observations. Generally, high-order dependence among labels and state-state transition with observation in the linear chain also can be considered.

4.3. Inference. Given a training dataset $\mathcal{D} = \{(\mathbf{o}_i, \mathbf{y}_i)\}_{i=1}^n$, we can apply m templates to obtain a new form of data with feature representation $\Phi(\mathbf{o}_i, \mathbf{y}_i) = [\Phi_1(\mathbf{o}_i, \mathbf{y}_i); \dots; \Phi_m(\mathbf{o}_i, \mathbf{y}_i); \Psi(\mathbf{y}_i)]$, $\forall i = 1, \dots, n$. There is an implicit group Ψ for the label sequence to capture the label dependence between the current label and its neighbors. Before directly using Algorithm 1 to solve this task, we need to solve Problem (12) as

$$(19) \quad \begin{aligned} \hat{\mathbf{y}}_i &= \arg \max_{\mathbf{y}'_i \in \mathcal{Y}^{l_i}} \Delta(\mathbf{y}_i, \mathbf{y}'_i) + \mathbf{w}^T \Phi(\mathbf{o}_i, \mathbf{y}'_i) \\ &= \arg \max_{\mathbf{y}'_i \in \mathcal{Y}^{l_i}} \Delta(\mathbf{y}_i, \mathbf{y}'_i) + \sum_{j=1}^m \langle \bar{\mathbf{w}}_j, \Phi_j(\mathbf{o}_i, \mathbf{y}'_i) \rangle + \langle \tilde{\mathbf{w}}, \Psi(\mathbf{y}'_i) \rangle. \end{aligned}$$

where $\mathbf{w} = [\bar{\mathbf{w}}_1; \dots; \bar{\mathbf{w}}_m; \tilde{\mathbf{w}}]$. We notice that given an observation sequence \mathbf{o} , the prediction function in Problem (5) is exactly the Problem (19) without the loss term $\Delta(\mathbf{y}_i, \mathbf{y}'_i)$. According to Corollary 23 in [30], we can derive the following equations,

$$(20) \quad \begin{aligned} \langle \bar{\mathbf{w}}_j, \Phi_j(\mathbf{o}_i, \mathbf{y}'_i) \rangle &= \sum_{t=1}^{l_i} \langle \bar{\mathbf{w}}_j, \phi_j(t, \mathbf{o}_i) \otimes \Lambda(y'_{i,t}) \rangle \\ &= \sum_{t=1}^{l_i} \sum_{\sigma \in \mathcal{Y}} \langle \bar{\mathbf{w}}_{j,\sigma}, \phi_j(t, \mathbf{o}_i) \rangle \delta(\sigma, y'_{i,t}) \end{aligned}$$

$$(21) \quad \begin{aligned} \langle \tilde{\mathbf{w}}, \Psi(\mathbf{y}'_i) \rangle &= \sum_{t=2}^{l_i} \langle \tilde{\mathbf{w}}, \Lambda(y'_{t-1}) \otimes \Lambda(y'_t) \rangle \\ &= \sum_{t=2}^{l_i} \sum_{\sigma \in \mathcal{Y}} \sum_{\sigma' \in \mathcal{Y}} \tilde{w}_{\sigma,\sigma'} \delta(\sigma, y'_{i,t-1}) \delta(\sigma', y'_{i,t}) \end{aligned}$$

By substituting (20) and (21) into Problem (19), we obtain

$$(22) \quad \langle \mathbf{w}, \Phi(\mathbf{o}_i, \mathbf{y}'_i) \rangle = \sum_{\sigma \in \mathcal{Y}} v_\sigma + \sum_{\sigma \in \mathcal{Y}} \sum_{\sigma' \in \mathcal{Y}} v_{\sigma,\sigma'}$$

where $v_\sigma = \sum_{j=1}^m \sum_{t=1}^{l_i} \langle \bar{\mathbf{w}}_{j,\sigma}, \phi_j(t, \mathbf{o}_i) \rangle \delta(\sigma, y'_{i,t})$ and $v_{\sigma,\sigma'} = \sum_{t=2}^{l_i} \tilde{w}_{\sigma,\sigma'} \delta(\sigma, y'_{i,t-1}) \delta(\sigma', y'_{i,t})$. Problem (5) with sequence feature representation (20) and (21) can be readily solved by the Viterbi algorithm [32]. In order to solve Problem (19), we need to design loss function $\Delta(\mathbf{y}_i, \mathbf{y}'_i)$ which is usually defined as Hamming loss, i.e. $\Delta(\mathbf{y}_i, \mathbf{y}'_i) = \sum_{t=1}^{l_i} (1 - \delta(y_{i,t}, y'_{i,t})) = l_i - \sum_\sigma u_\sigma$ where $u_\sigma = \sum_{t=1}^{l_i} \delta(y_{i,t}, \sigma) \delta(y'_{i,t}, \sigma)$. Therefore, $\Delta(\mathbf{y}_i, \mathbf{y}'_i)$ can be merged into the first term in the right-hand side of (22), so the Viterbi algorithm can be used to find the most violated constraints.

4.4. Experiments. In this section, we evaluate our proposed method called Multiple Template Learning for sequence labeling (MTL^{hmm}) on some natural language processing tasks, which could be transformed as the sequence labeling problem, and compare it with the state-of-the-art sequence labeling algorithms such as CRF with a Gaussian prior over parameters ($\text{CRF}(\ell_2)$), CRF with a Laplacian prior ($\text{CRF}(\ell_1)$) and Structural SVMs for sequence learning (SVM^{hmm}). The implementations used for comparison are CRF++ software¹ and SVM^{hmm} software². We implement MTL^{hmm} in C++. For fair comparisons, we use the same feature templates from the information provided by training dataset without using any external prior knowledge. The set of templates is also defined arbitrarily without considering any language prior knowledge. All templates used in this paper only capture the local context information by uni-gram, bi-gram or tri-gram. Although the templates listed in the following sections may not be the state of the art, the goal of this paper is to examine whether or not the weighted group features can achieve better performance than the uniform weighting strategy adopted in CRF and SVM^{hmm} . Users can add more templates in terms of their own prior knowledge about the specific task if it is needed.

The template format used in sequence learning problem is borrowed from CRF++. Before applying templates to datasets (including training dataset and test dataset), we preprocess them to be multiple tokens. In addition, each token consists of multiple columns with a fixed number. The definition of the token depends on the task. For Chinese word segmentation, one token is one Chinese character in a sentence, while one word is a token for Chinese part-of-speech tagging and named entity recognition. Each column of a token represents some kinds of semantic meaning, such as word and part-of-speech tag in the named entity recognition task. In each template, special macro %x[row,col] will be used to specify a token in the input data. row specifies the relative position from the current focusing token and col specifies the absolute position of the column. The template index is used to maintain the relative position of features. For the bi-gram features, we have the different meaning with CRF++ definition. In this paper, bi-gram features combine token information in two relative positions, while CRF++ denotes it as the state transition feature. We also use B to denote the state-state transition features in the form of (16).

In order to compare different algorithms fairly, we use the features generated by CRF++ from a given template set as the input for SVM^{hmm} and MTL^{hmm} . For CRF++, we use the default setting for all the parameters except the parameter σ in CRF++ toolbox which is tuned in the range $[10^{-1}, 10^0, 10^1, 10^2]$ by command "-c". Command parameter "-a" is used to determine whether the $\text{CRF}(\ell_2)$ or $\text{CRF}(\ell_1)$

¹<http://crfpp.sourceforge.net/>

²http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html

type	Index	Template
uni-gram	U00	%x[-2,0]
	U01	%x[-1,0]
	U02	%x[0,0]
	U03	%x[1,0]
	U04	%x[2,0]
bi-gram	U08	%x[-1,0]/%x[0,0]
	U09	%x[0,0]/%x[1,0]
	U10	%x[-2,0]/%x[-1,0]
	U11	%x[1,0]/%x[2,0]
	U12	%x[-2,0]/%x[0,0]
	U13	%x[-1,0]/%x[1,0]
	U14	%x[0,0]/%x[2,0]
	U15	%x[-2,0]/%x[1,0]
	U16	%x[-1,0]/%x[2,0]
tri-gram	U05	%x[-2,0]/%x[-1,0]/%x[0,0]
	U06	%x[-1,0]/%x[0,0]/%x[1,0]
	U07	%x[0,0]/%x[1,0]/%x[2,0]
	B	state transition

TABLE 1. The set of templates used for Chinese word segmentation task.

Training Dataset	# Sentence	#Token
AS	708,953	8,368,050
CityU	53,019	2,403,355
PKU	19,056	1,826,458
MSR	86,924	4,050,469

TABLE 2. Training dataset for Chinese word segmentation task.

is used. The default setting of CRF++ does not remove any features according to the occurred frequency in the training data. Accordingly, in addition to the default setting of SVM^{hmm} , we tune C in the range of $n \times [10^{-1}, 10^0, 10^1, 10^2]$ to obtain the equivalent setting and use the recommended termination condition parameter setting with “-e 0.5” from SVM^{hmm} website. The proposed algorithm SVM^{hmm} has the similar setting with SVM^{hmm} , but we also set the maximum iteration to 500. The best performance for each algorithm is reported by tuning their own parameters.

In the next subsections, we will show the detailed experiments on the natural language processing tasks, such as Chinese Word Segmentation, Chinese Part-of-Speech tagging and Named Entity Recognition. The same experimental setting are applied to all tasks.

4.4.1. Chinese Word Segmentation. Chinese is written without inter-word spaces, such as the blank character in English, so it is very hard to find the word boundaries. However, this process is an essential first step in many natural language processing applications such as mono- and cross-lingual information retrieval and text-to-speech systems [9]. Figure 1 gives an example of Chinese word segmentation task where Figure 1(a) shows the input Chinese character sequence, and Figure

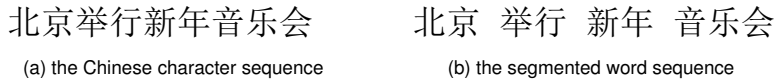


FIGURE 1. An example for Chinese word segmentation.

1(b) shows the segmented word sequence with blank characters inserted into the proper position of original character sequence.

Chinese word segmentation problem has been an active area of research in computational linguistics for several decades. Many algorithms are invented to address this problem in the literature. In this paper, we mainly follow the principle of the character tagging for Chinese word segmentation [35], especially for sequence character tagging methods such as CRFs [23], which has been considered as the state-of-the-art algorithm. In the character tagging principle, each character is attached by a specific encoded class label. The label sequence can be used to recover the segmented word sequence. The detailed explanation of the encoding labels can be observed in the following part of this section.

The second International Chinese Word Segmentation Bakeoff task provides the platform to evaluate different methods. It contains four corpus, such as Academia Sinica (AS), City University of Hong Kong (CityU), Peking University (PKU) and Microsoft Research (MSR), which are available from website ³. Each corpus is comprised of a training dataset and a ground-truth testing dataset. The detailed corpus information in Table 2 is referred to [9]. All the algorithms are trained on the training datasets, and then results are reported on the testing dataset by removing the label from the ground-truth dataset and evaluated by the following measures: test recall (Recall), test precision (Prec), balanced F score ($F1 = 2 \times P \times R / (P + R)$), and recall on in-vocabulary words (R_{iv}). All the measure scores are obtained by the attached scoring script along the corpus.

As we discussed in the previous sections, all the methods mentioned above need feature templates to extract the features from labeled corpus. The set of templates used for Chinese word segmentation task is shown in Table 1. We use three label notations such as B, I, E (B is the beginning character of a word; I is the inner character of a word; E is the end character of a word) to transform Chinese word segmentation task into a label sequence learning problem.

To justify whether improper or redundant templates could affect sequence prediction performance or not, we divided the set of templates into two categories: TP1 and TP2. TP1 includes the templates indexed from $U00$ to $U09$ plus B ; while TP2 includes the ones indexed from $U00$ to $U16$ plus B . In TP2, we deliberately enumerate all the possible bi-gram templates in the window size $[-2, +2]$. This construction of templates has a high probability to include redundant information. We run all the algorithms on the features extracted from the training dataset by applying TP1 and TP2, respectively.

The testing results on the above two different set of templates are reported in Table 3. We observe the following facts. In TP1, four algorithms have similar F1 score with a difference at most 0.5%, except that $CRF(\ell_2)$ is 1% higher than SVM^{hmm} on AS. However, MTL^{hmm} always demonstrate the highest R_{iv} , which means weighting strategy is more useful to predict the known words correctly. Even

³<http://www.sighan.org/bakeoff2005/>

Dataset	Algorithm	Recall	Prec	F1	R_{iv}
AS	CRF (ℓ_2)	95.6 / 95.1	94.4 / 93.8	95.0 / 94.4	97.0 / 96.4
	CRF (ℓ_1)	95.4 / 95.0	94.1 / 93.4	94.8 / 94.2	96.7 / 96.4
	SVM ^{hmm}	94.6 / 94.3	93.2 / 93.3	93.9 / 93.8	96.0 / 95.6
	MTL ^{hmm}	95.6 / 95.8	93.9 / 93.9	94.7 / 94.8	97.2 / 97.3
CityU	CRF (ℓ_2)	94.1 / 92.8	94.3 / 92.6	94.2 / 92.7	96.3 / 95.1
	CRF (ℓ_1)	93.4 / 92.9	93.6 / 92.9	93.5 / 92.9	95.6 / 95.1
	SVM ^{hmm}	94.3 / 94.1	94.1 / 93.5	94.2 / 93.8	96.4 / 96.4
	MTL ^{hmm}	95.2 / 95.2	94.2 / 94.3	94.7 / 94.8	97.6 / 97.6
PKU	CRF (ℓ_2)	92.7 / 92.2	94.2 / 93.3	93.4 / 92.7	94.7 / 94.2
	CRF (ℓ_1)	91.6 / 91.8	93.1 / 92.4	92.4 / 91.7	93.8 / 93.4
	SVM ^{hmm}	92.6 / 93.0	93.7 / 93.4	93.2 / 93.2	94.9 / 95.0
	MTL ^{hmm}	93.8 / 94.0	93.3 / 93.6	93.6 / 93.8	96.1 / 96.2
MSR	CRF (ℓ_2)	96.5 / 95.7	96.7 / 96.0	96.6 / 95.8	97.3 / 96.5
	CRF (ℓ_1)	96.2 / 95.7	96.2 / 95.6	96.2 / 95.6	96.9 / 96.5
	SVM ^{hmm}	96.7 / 96.1	96.5 / 96.5	96.6 / 96.3	97.6 / 96.7
	MTL ^{hmm}	96.9 / 97.0	96.5 / 96.4	96.7 / 96.7	97.9 / 98.0

TABLE 3. The Chinese word segmentation results of different algorithms on all the testing datasets. The left-hand side of / is the results using TP1, while the right-hand side is using TP2.

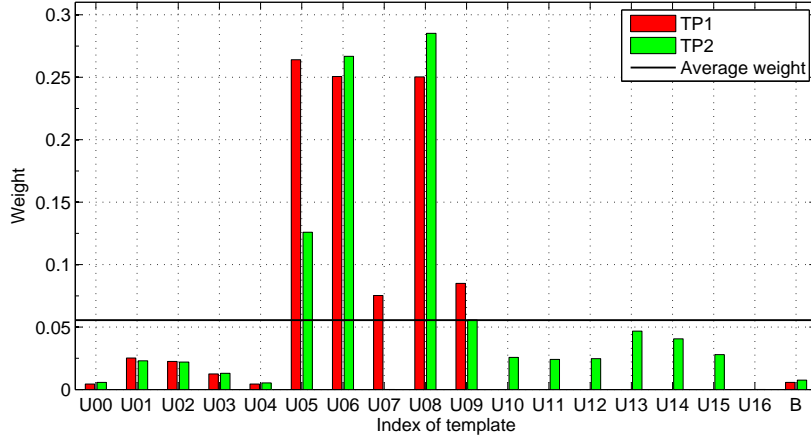
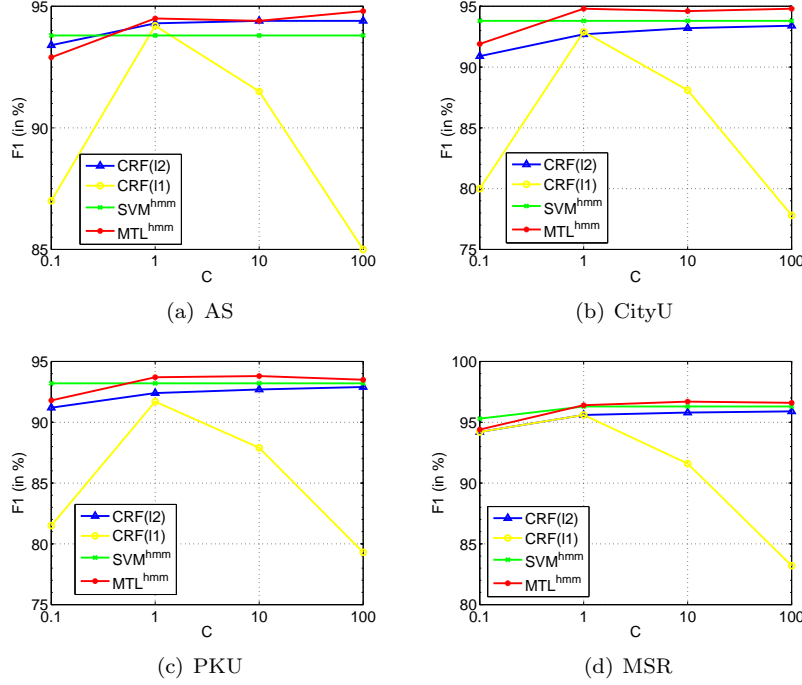


FIGURE 2. The learned template weights by MTL^{hmm} for Chinese word segmentation in TP1 and TP2, respectively, and the solid line for average weight.

though MTL^{hmm} and SVM^{hmm} use the same set of features, MTL^{hmm} is better than SVM^{hmm} according to F1 score. This shows that the proposed weighting template strategy is helpful to boost the performance.

In TP2, the similar phenomena could be observed, but the difference of F1 score among different methods are enlarged. For example, for CityU dataset, MTL^{hmm} is 2.1% higher than CRFs. MTL^{hmm} consistently shows the best F1 score among four

FIGURE 3. F1 measure in terms of varied C in TP2.

algorithms. By comparing the results between TP1 and TP2, we find that adding more templates, such as templates U_{10} - U_{16} , degrades the performance of CRFs and SVM^{hmm} , but MTL^{hmm} obtains slight improvements. Therefore, the performance of MTL^{hmm} is not greatly affected by the arbitrary added templates. This may be due to the fact that MTL^{hmm} can effectively identify the important templates from a given set of arbitrary templates, and can remove unimportant templates simultaneously. However, $CRF(\ell_2)$ and SVM^{hmm} do not consider this information, and $CRF(\ell_1)$ works poorly on CityU and PKU. This statement can be justified by the learned template weights. Figure 2 shows one example of the learned template weights on AS with the settings of TP1 and TP2. We can observe that most of the weights are assigned to templates U_{05} to U_{09} in TP1, but U_{07} turns out to be not important when adding U_{10} - U_{16} to TP1. Moreover, U_{16} is also considered as a redundant template in TP2. We can conclude that MTL^{hmm} can effectively and efficiently handle arbitrary templates without performance degradation, but CRFs and SVM^{hmm} cannot.

In addition, Figure 3 shows the variations of testing F1 measure over the range of C used in the experimental setting. MTL^{hmm} obtain better results than the rest of algorithms in the range with large C , i.e. $C \geq 1$ in Figure 3. Therefore, weighting groups of features can boost the F1 measure on all four Chinese word segmentation datasets. $CRF(\ell_1)$ varies greatly in terms of different C , which implies that aggressively removing features could degrade prediction performance.

北京/ns 举行/v 新年/t 音乐会/n

FIGURE 4. An example for Chinese part of speech tagging.

4.4.2. *Chinese Part-of-Speech Tagging.* Part-of-speech (POS) tagging is a task to assign a lexical category or part-of-speech tag to each word in a sentence. POS is an important component in nature language processing such as shallow parsing or full parsing. Figure 4 shows an example where *ns* is location name, *v* is a verb, *t* is a time word and *n* is a noun word.

Similar to Chinese word segmentation, part-of-speech tagging could also be transformed as a sequence labeling task based on words in English [13, 1, 12]. Chinese part-of-speech tagging is more difficult to be solved due to the implicit features, not like the prefixes and suffixes in English which are more explicit to determine the part-of-speech for each word in a sentence. The only information that we can use is the character or word. The property of sequence learning methods based on multiple interacting features or long-range dependencies of the observations can improve hidden Markov model or Maximum Entropy model [26].

type	Index	Template
uni-gram	U00	%x[-2,0]
	U01	%x[-1,0]
	U02	%x[0,0]
	U03	%x[1,0]
	U04	%x[2,0]
bi-gram	U05	%x[-2,0]/%x[-1,0]
	U06	%x[1,0]/%x[2,0]
tri-gram	U07	%x[-1,0]/%x[0,0]/%x[1,0]
	B	state transition

TABLE 4. The set of templates used for Chinese Part-Of-Speech task.

In order to compare our proposed algorithm with CRFs and SVM^{hmm} on Chinese part-of-speech task, we evaluate on the first month of Peking people’s daily newspaper in 1998, which is available from www.icl.pku.edu.cn. This dataset consists of 19,484 sentences. For simplicity, we divide the whole data into training dataset and test dataset to be equal number of sentences in the original sentence order. For training data, there are 9,746 sentences with total 567,886 tokens. There are 44 part-of-speech tags. However, tag *Yg* only appears in the test dataset, so there are 43 tags that we really consider to train a part-of-speech tagging model. Since all the methods we mentioned in this paper can deal with complex features from observations, we use the feature templates in the Table 4. Here, each token is a Chinese word. We consider this set of templates as an instance for comparing different algorithms.

Method	CRF (ℓ_2)	CRF (ℓ_1)	SVM^{hmm}	MTL^{hmm}
Accuracy	92.45	92.59	93.14	93.59

TABLE 5. The accuracy for Chinese Part-of-Speech task.

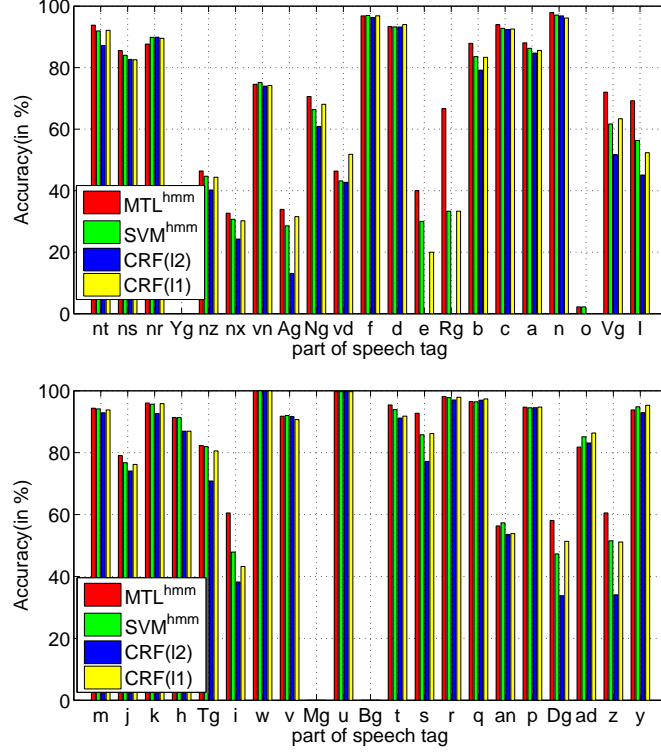


FIGURE 5. The test accuracy of different algorithms on individual part-of-speech tags.

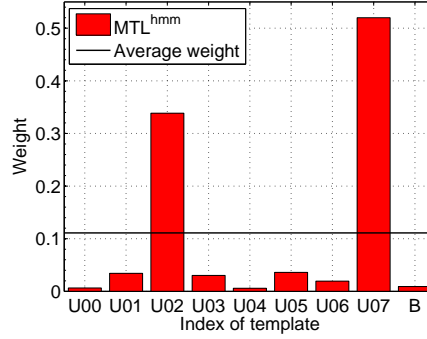


FIGURE 6. The learned template weights by MTL^{hmm} for Chinese part-of-speech tagging and the solid line for the average weight.

All algorithms can achieve the best performance with parameters $\sigma = 100$ and $C = 100n$, except $\sigma = 1$ for $CRF(\ell_1)$. Table 5 shows the accuracy of each algorithm. From Table 5, we can observe that our proposed method can achieve higher accuracy than $CRF(\ell_2)$, $CRF(\ell_1)$ and SVM^{hmm} . For analysis of the results in details, we plot the accuracy bars of all algorithms in terms of part-of-speech tag in Figure 5.

Except that Yg is not recognized because it does not appear in training dataset, all the algorithm cannot identify Mg and Bg . Regardless of these difficult tags which need more sophisticated templates designed from domain experts, $\text{CRF}(\ell_2)$ also cannot identify tags, such as e , Rg and o , but SVM^{hmm} and MTL^{hmm} can. Roughly, $\text{CRF}(\ell_2)$ is perform worse than SVM^{hmm} and MTL^{hmm} on almost all the tags from Figure 5. Therefore, we can argue that sequence learning based on large margin principle has much more advantages than $\text{CRF}(\ell_2)$ in this task.

Even though both SVM^{hmm} and MTL^{hmm} are based on large margin principle and use same features, their performances are slightly different. MTL^{hmm} performs better or comparable with SVM^{hmm} . Especially for tags e , Rg , Vg , I , i , Dg , z , MTL^{hmm} obtain great improvement over SVM^{hmm} . This observation implies that MTL^{hmm} learns the weight of each template can improve the performance of the part-of-speech tagging, and also prove that our initial assumption is reasonable. We also report the template weights we learned in Figure 6, where average templates are used to represent SVM^{hmm} and CRFs since they do not learn the weights. From Figure 6, we can clearly see that the templates $U02$ and $U07$ dominates the weight vectors. It means that $U02$ and $U07$ are very important templates for part-of-speech tagging task. It is very intuitive to explain this phenomenon since the features generated by $U02$ means the current word can indicate its part-of-speech tag and $U07$ means the nearest neighbors are the most important indicators. From the results shown in Table 5 and Figure 5, we can argue that the weighted template schema is really helpful to boost the performance of part-of-speech tagging task.

type	Index	Template
uni-gram	U00	%x[-2,0]
	U01	%x[-1,0]
	U02	%x[0,0]
	U03	%x[1,0]
	U04	%x[2,0]
	U10	%x[-2,1]
	U11	%x[-1,1]
	U12	%x[0,1]
	U13	%x[1,1]
	U14	%x[2,1]
bi-gram	U05	%x[-1,0]/%x[0,0]
	U06	%x[0,0]/%x[1,0]
	U15	%x[-2,1]/%x[-1,1]
	U16	%x[-1,1]/%x[0,1]
	U17	%x[0,1]/%x[1,1]
	U18	%x[1,1]/%x[2,1]
tri-gram	U20	%x[-2,1]/%x[-1,1]/%x[0,1]
	U21	%x[-1,1]/%x[0,1]/%x[1,1]
	U22	%x[0,1]/%x[1,1]/%x[2,1]
	B	state transition

TABLE 6. The set of templates used for Named Entity Recognition task

4.4.3. *Language-Independent Named Entity Recognition.* Named entities are phrases that contain the names of persons, organizations, locations, times and quantities.

Training Dataset	# Sentence	#Token
Spanish (ESP)	8,323	264,715
Dutch (NED)	15,806	202,931

TABLE 7. Training dataset for Named Entity Recognition task.

For example, a sentence “*U. N. official Ekeus heads for Baghdad.*” can be tagged by phrases “[*ORG U.N.*] [*official*] [*PER Ekeus*] [*heads for*] [*LOC Baghdad*].” where ORG is an organization name, PER is a person name and LOC is a location name. In this task, the goal is to identify all the specific types of entities in a given input sentence.

Sequence tagging models, such as CRFs [16], have been used for name entities recognition task and demonstrate promising results. We evaluate our method and other methods on the CoNLL-2002 language-independent named entity recognition shared task, which is available from website ⁴. The data from the shared task of CoNLL-2002 consists of three files per language: one training file and two test files testa and testb. The first test file will be used in the development phase for finding good parameters for the learning system. The second test file will be used for the final evaluation. These data files are available for two languages: Spanish (ESP) and Dutch (NED). Four types of phrases are defined: person names (PER), organizations (ORG), locations (LOC) and miscellaneous names (MISC). The label notation uses B, I, O format for each entity where B is the start word of an entity, I is the inner word of an entity, and O is a word not belonging to any entity. Table 7 shows the training data description in details. Moreover, each token is assigned a part-of-speech (POS) tag. The set of templates used for named entity recognition task are defined in Table 6 where some templates include POS tag information, such as *U10* using the POS tag two tokens before.

Table 8 shows the experimental results evaluated on individual validation datasets and test datasets, respectively. From Table 8, we can observe that MTL^{hmm} significantly outperforms SVM^{hmm} and CRFs in terms of recall and F-value on both Spanish and Dutch datasets. Notably, the precision of MTL^{hmm} on Dutch data is even better than that of CRFs by 4% as well. The overall performance could be observed from Table 8 in details. On Spanish dataset, MTL^{hmm} can achieve higher F1 score than others in terms of PER and MISC, and is comparable on the recognition of ORG and PER with the best F1 score of others. On Dutch dataset, the improvement is more obvious. Except for PER, MTL^{hmm} performs much better than others in terms of F1. This shows the effectiveness of MTL^{hmm} on the task of named entity recognition. We also show the weights we learned on Spanish and Dutch dataset, respectively. Even though they are different languages, the dominated templates are similar, that is *U02*, *U05* and *U06*. It means that the current word features (*U02*) and the bi-gram features around the current word (*U05* and *U06*) are the most important features to determine entities in the given set of templates.

⁴<http://www.cnts.ua.ac.be/conll2002/ner/>

Spanish (ESP)	Methods	Validation set			Test set		
		Prec	Recall	F1	Prec	Recall	F1
LOC	CRF (ℓ_2)	72.95	73.10	73.02	84.14	66.05	74.01
	CRF (ℓ_1)	71.20	71.78	71.49	82.86	64.67	72.64
	SVM ^{hmm}	67.05	76.24	71.35	78.36	67.80	72.70
	MTL ^{hmm}	68.59	77.36	72.71	80.70	69.83	74.88
MISC	CRF (ℓ_2)	60.35	30.79	40.77	64.06	36.18	46.24
	CRF (ℓ_1)	58.46	34.16	43.12	57.48	36.18	44.40
	SVM ^{hmm}	54.52	39.33	45.69	56.56	40.59	47.26
	MTL ^{hmm}	61.57	37.08	46.28	62.10	40.00	48.66
ORG	CRF (ℓ_2)	81.39	62.00	70.38	81.63	70.79	75.82
	CRF (ℓ_1)	79.64	59.35	68.01	82.20	69.93	75.57
	SVM ^{hmm}	78.39	60.18	68.09	78.17	69.57	73.62
	MTL ^{hmm}	77.85	62.24	69.17	78.25	73.00	75.54
PER	CRF (ℓ_2)	83.15	60.97	70.35	84.02	77.96	80.88
	CRF (ℓ_1)	83.39	62.03	71.14	82.87	77.69	80.20
	SVM ^{hmm}	84.50	64.24	72.99	85.53	80.41	82.89
	MTL ^{hmm}	88.01	59.49	71.00	88.46	77.14	82.41
Overall	CRF (ℓ_2)	78.00	61.03	68.48	81.76	67.52	73.96
	CRF (ℓ_1)	76.58	60.34	67.50	80.75	66.70	73.00
	SVM ^{hmm}	74.39	62.82	68.12	78.17	68.50	73.02
	MTL ^{hmm}	76.09	62.32	68.52	79.96	69.74	74.50

Dutch (NED)	Methods	Validation set			Test set		
		Prec	Recall	F1	Prec	Recall	F1
LOC	CRF (ℓ_2)	82.52	49.27	61.70	82.50	55.43	66.31
	CRF (ℓ_1)	82.71	45.93	59.06	80.91	55.30	65.69
	SVM ^{hmm}	70.94	51.98	60.00	76.21	58.79	66.37
	MTL ^{hmm}	88.76	49.48	63.54	92.88	62.40	74.65
MISC	CRF (ℓ_2)	90.46	36.76	52.28	89.57	38.33	53.69
	CRF (ℓ_1)	86.55	41.31	55.93	86.03	42.54	56.93
	SVM ^{hmm}	76.86	38.64	51.42	79.15	40.94	53.97
	MTL ^{hmm}	86.71	47.99	42.42	87.58	48.69	62.59
ORG	CRF (ℓ_2)	88.51	33.67	48.79	87.60	38.44	53.43
	CRF (ℓ_1)	82.89	36.01	50.20	83.18	39.80	53.83
	SVM ^{hmm}	83.77	32.36	46.69	81.49	35.94	49.88
	MTL ^{hmm}	83.62	42.42	56.29	85.01	46.94	60.48
PER	CRF (ℓ_2)	64.31	55.62	59.65	73.19	70.13	71.63
	CRF (ℓ_1)	66.49	53.06	59.02	74.70	67.21	70.76
	SVM ^{hmm}	60.28	55.48	57.78	69.07	68.12	68.59
	MTL ^{hmm}	69.77	52.20	59.72	78.71	64.30	70.78
Overall	CRF (ℓ_2)	77.66	43.31	55.61	80.79	50.57	62.20
	CRF (ℓ_1)	77.53	43.92	56.08	80.05	51.31	62.53
	SVM ^{hmm}	70.16	43.96	54.05	74.77	50.90	60.57
	MTL ^{hmm}	80.64	47.94	60.13	85.06	55.34	67.06

TABLE 8. The named entity recognition average performance of different algorithms on Spanish and Dutch datasets, respectively.

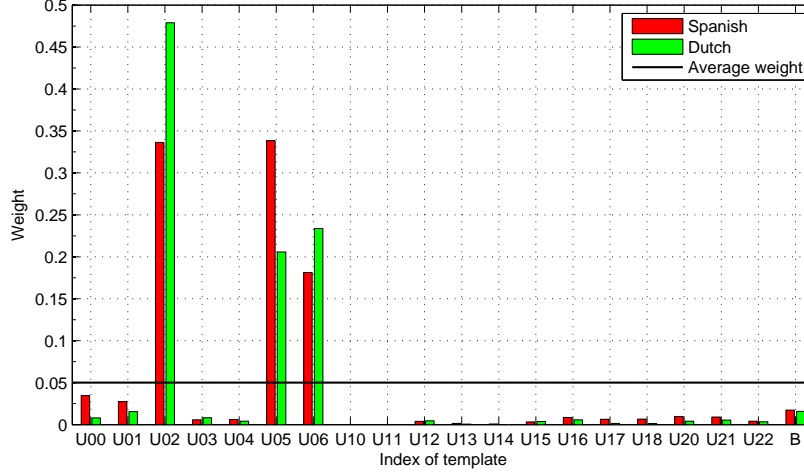


FIGURE 7. The learned template weights by MTL^{hmm} on Spanish and Dutch, respectively, and the solid line for the average weight

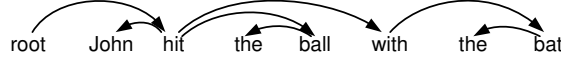


FIGURE 8. An example dependency tree (adapted from [18]).

5. MULTIPLE TEMPLATE LEARNING FOR DEPENDENCY PARSING

In this section, we apply the proposed multiple template learning framework to dependency parsing task where the output structure is a directed tree structure. Before directly using Algorithm 1 to solve dependency parsing problem, we first formulate it as the multiple template learning problem by defining compatibility function over tree structure and the group features from a predefined templates. Finally, we give the feasible inference methods for finding most violated constraints, as well as predicting unknown inputs.

5.1. Dependency Parsing. The goal of dependency parsing is to find a generic dependency tree $\mathcal{T}_{\mathbf{x}}$ for a given generic input sentence $\mathbf{x} = (x_1, \dots, x_l)$ with l tokens. One example is shown in Figure 8 with an input sentence "John hit the ball with the bat". An additional input node called "root" are manually introduced as the root node of the dependency tree. The input sentence can be represented by augmenting root node to be $\mathbf{x} = (x_0, x_1, \dots, x_l)$ where $x_0 = \text{root}$. The dependency tree is a tree structure with directed edges over the input nodes.

[18] showed that dependency parsing can be formulated as the search for a maximum spanning tree in a directed graph. The generic dependency tree can be represented as a directed graph $\mathcal{T}_{\mathbf{x}} = (\mathcal{V}_{\mathbf{x}}, \mathcal{E}_{\mathbf{x}})$ by its vertex set $\mathcal{V}_{\mathbf{x}} = \{x_0, x_1, \dots, x_l\}$ and directed edge set $\mathcal{E}_{\mathbf{x}} \subseteq \{(u, v) : u \neq v, (u, v) \in [0 : l] \times [1 : l]\}$ under the following constraints that $\mathcal{T}_{\mathbf{x}}$ is weakly connected and all the nodes in $\mathcal{V}_{\mathbf{x}}$ have an in-degree of exactly one except the unique root node with in-degree zero. The constraints

actually make a dependency graph be a dependency tree structure. There exist two types of dependency trees. One is called projective dependency tree (see Figure 8) in which a word and its descendants form a contiguous substring of a sentence. If the sentence with a root as the first word in their linear order, the edges drawn above the words have no crossings. If there is at least one crossing in a tree, it is called non-projective dependency tree.

The goal of dependency parsing is to learn the hypotheses

$$(23) \quad f : \mathbb{X} \rightarrow \mathbb{T},$$

where \mathbb{X} is the domain of generic input sentences, and \mathbb{T} is the set of feasible dependency trees in \mathbb{X} . The score function $s : \mathbb{X} \times \mathbb{T} \rightarrow \mathbb{R}$ is defined to be the compatibility function, so that we can obtain the hypotheses by maximizing the scores over all the feasible trees of the input sentence \mathbf{x} ,

$$(24) \quad f(\mathbf{x}) = \arg \max_{\mathcal{T}_{\mathbf{x}} \in \mathbb{T}} s(\mathbf{x}, \mathcal{T}_{\mathbf{x}}).$$

The score functions are usually parameterized functions over a set of features which are extracted from training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{T}_{\mathbf{x}_i})\}_{i=1}^n$. Next, we explain the detailed definition of score function on tree structures and the corresponding feature representation.

5.2. Feature Representation. By factoring the score of the tree $\mathcal{T}_{\mathbf{x}}$ into the sum of edge scores [8], we have made dependency parsing equivalent with finding maximum spanning trees. Given an edge $(u, v) \in \mathcal{E}_{\mathbf{x}}$ in the dependency tree $\mathcal{T}_{\mathbf{x}}$, the score of this edge could be defined as the inner product between a high dimensional feature representation $\bar{\phi}(u, v, \mathbf{x})$ of the edge and a weight vector \mathbf{w} , that is the score function $s(u, v, \mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \bar{\phi}(u, v, \mathbf{x}) \rangle$. Correspondingly, the score of the dependency tree $\mathcal{T}_{\mathbf{x}}$ is

$$(25) \quad s(\mathcal{T}_{\mathbf{x}}, \mathbf{x}; \mathbf{w}) = \sum_{(u,v) \in \mathcal{E}_{\mathbf{x}}} s(u, v, \mathbf{x}; \mathbf{w}) = \sum_{(u,v) \in \mathcal{E}_{\mathbf{x}}} \langle \mathbf{w}, \bar{\phi}(u, v, \mathbf{x}) \rangle.$$

Assuming that the feature representation $\bar{\phi}$ and the weight vector \mathbf{w} are given, dependency parsing can be considered to solve Problem (24).

Since the feature representation $\bar{\phi}(u, v, \mathbf{x})$ is formulated on the position u, v and the input sentence \mathbf{x} , we can define templates to extract overlapping features on the whole observation sentence with the direction of attachment as well as the distance between x_u and x_v creating the dependency. The feature dimension of $\bar{\phi}$ is usually very high, but it is very sparse, so the sparse vector presentations are useful to reduce the calculation to linear in the number of active features for a given edge.

For intuitive understanding, one template can be defined as “*the words in u and v position*”. Given a sentence “*root John hit the ball with the bat*”, one of the features by applying the given template is

$$\phi(u = 2, v = 4) = \begin{cases} 1 & \text{if } x_u = \text{“hit” and } x_v = \text{“ball”} \\ 0 & \text{otherwise.} \end{cases}$$

Generally, more templates are needed to obtain high accuracy for dependency parsing. For instance, the features take the form of a POS trigram for all words linearly between the parent and the child, which is particularly helpful for nouns identifying their parents [17]. Similar to sequence learning problems, we can group the subset of

features together extracted from each template and form a group of features. Supposing that there are m templates, we can apply j^{th} template over each sentence-tree pair $(\mathbf{x}_i, \mathcal{T}_{\mathbf{x}_i})$ to construct a feature vector $\Phi_j(\mathbf{x}, \mathcal{T}_{\mathbf{x}_i}) = \sum_{(u,v) \in \mathcal{E}_{\mathbf{x}_i}} \bar{\phi}_j(u, v, \mathbf{x})$ where $\bar{\phi}_j(u, v, \mathbf{x})$ is the vector representation of all the features in j^{th} group after applying j^{th} template to \mathcal{D} . Therefore, we can obtain a group representation of features as $\Phi(\mathbf{x}, \mathcal{T}_{\mathbf{x}_i}) = [\Phi_1(\mathbf{x}, \mathcal{T}_{\mathbf{x}_i}); \dots; \Phi_m(\mathbf{x}, \mathcal{T}_{\mathbf{x}_i})]$.

5.3. Inference. Given the model parameter $\mathbf{w} = [\mathbf{w}_1; \dots; \mathbf{w}_m]$, we can predict the dependency parsing tree $\hat{\mathcal{T}}_{\mathbf{x}}$ by solving Problem (24) given an input sentence \mathbf{x} . According to the features defined in Section 5.2, the prediction problem can be rewritten as

$$\begin{aligned}
 (26) \quad \hat{\mathcal{T}}_{\mathbf{x}} &= \arg \max_{\mathcal{T}_{\mathbf{x}} \in \mathbb{T}} \langle \mathbf{w}, \Phi(\mathbf{x}, \mathcal{T}_{\mathbf{x}}) \rangle \\
 &= \arg \max_{\mathcal{T}_{\mathbf{x}} \in \mathbb{T}} \sum_{j=1}^m \langle \mathbf{w}_j, \Phi_j(\mathbf{x}, \mathcal{T}_{\mathbf{x}}) \rangle \\
 &= \arg \max_{\mathcal{T}_{\mathbf{x}} \in \mathbb{T}} \sum_{j=1}^m \sum_{(u,v) \in \mathcal{E}_{\mathbf{x}}} \langle \mathbf{w}_j, \bar{\phi}_j(u, v, \mathbf{x}) \rangle
 \end{aligned}$$

According to [18], there are two algorithms which can solve Problem (27) in terms of two types of dependency trees, respectively. For projective dependency tree, Eisner algorithm [8] has a runtime of $O(l^3)$, while Chu-Liu-Edmonds [6] provides non-projective parsing complexity $O(l^2)$ where l is the length of the input sentence.

The most violated constraint should be the constraint with the dependency tree $\hat{\mathcal{T}}_{\mathbf{x}_i}$ such that $\forall i = 1, \dots, n$

$$\begin{aligned}
 (27) \quad \hat{\mathcal{T}}_{\mathbf{x}_i} &= \arg \max_{\mathcal{T}_i \in \mathbb{T}} \Delta(\mathcal{T}_{\mathbf{x}_i}, \mathcal{T}_i) - \mathbf{w}^T \delta \Phi^i(\mathcal{T}_i) \\
 &= \arg \max_{\mathcal{T}_i \in \mathbb{T}} \Delta(\mathcal{T}_{\mathbf{x}_i}, \mathcal{T}_i) + \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathcal{T}_i) \rangle.
 \end{aligned}$$

The loss for dependency parsing is defined as the number of words that have the incorrect parent, i.e. $\Delta(\mathcal{T}_{\mathbf{x}_i}, \mathcal{T}_i) = \sum_{v=0}^{l_i} \sum_{(u,v) \in \mathcal{E}_{\mathbf{x}_i}} \sum_{(u',v) \in \mathcal{E}_i} (1 - \delta(u, u'))$, where x_u is the parent node, and x_v is the child node. The inference algorithms are employed to solve Problem (27) as well. This is due to the decomposition property of loss function which is closely related to Hamming loss used in the sequence learning.

5.4. Experiments. We evaluate MTL for dependency parsing by comparing with MSTParser software⁵, which implements an online algorithm for dependency parsing [17]. We adapt MSTParser for multiple template learning in Java called MTL-Parser for projective dependency parsing. The comparison is mainly to examine whether multiple template learning algorithms can boost the performance by learning the weight of each template simultaneously. We perform comparisons on CoNLL-X shared task: Multi-lingual Dependency Parsing. This shared task consists of datasets from 13 different languages, in our experiments, we choose two datasets: Danish and Swedish, which are available from website⁶. For each language, the dataset consists of two data files. One is the training dataset, while

⁵<http://sourceforge.net/projects/mstparser/>

⁶http://nextens.uvt.nl/~conll/post_task_data.html

Dataset	#Features	#Templates	MSTParser(%)		MTLParser(%)	
			Accuracy	Complete	Accuracy	Complete
Danish	2,137,252	568	86.79	30.12	86.86	32.61
Swedish	2,292,216	612	83.59	38.56	83.95	40.62

TABLE 9. The performance of dependency parsing on two datasets. *#Features* is the number of features used in both MSTParser and MTLParser. *#template* is the number of templates naturally formed groups over all features in MTLParser. *Accuracy* is the number of words that correctly identified their parents in the tree. *Complete* is the number of sentences for which the entire dependency tree are correct.

the other is the ground truth test dataset. The training dataset is used to train the prediction model, and then testing results are reported on the ground truth test dataset. In each dataset, sentences separated by a blank line. A sentence consists of at least one token, and each token consists of ten fields, but we only use the following fields: Token counter starting from 1 for each new sentence (ID), word/punctuation symbol (FORM), Lemma of word form (LEMMA), Coarse-grained POS Tag (CPOSTAG), Fine-grained POS tag (POSTAG), the head of dependency relation (HEAD).

We use the templates implemented in MSTParser. We take the following templates: (1) POS (including CPOSTAG and POSTAG) trigrams: the POS of the head, that of the modifier and that of a word in between, for all distinct POS tags for the words between the head and the modifier. Each relative position from the head to the modifier can be considered as a different type of template. (2) The form of POS 4-gram: The POS of the head, modifier, word before/after head and word before/after modifier. (3) Two items: each template consists of two observations, e.g. head word, head POS/LEMMA, child word, and child POS/LEMMA. All templates are conjoined with the direction of attachment as well as the distance between the two words creating the dependency. For the distance is longer than 10, it is set to 10; If between 10 and 5, it is 5, otherwise it does not change.

Table 9 shows the results of MSTParser and MTLParser on two language-independent dependency parsing datasets. We run MSTParser in different epochs ranging from 10 to 30, and report the best results, while the parameter C in MTLParser is set to $1000n$. We observe that MTLParser can obtain comparable results with MSTParser in terms of accuracy, but there is a great improvement more than 2% on complete correct dependency trees. This is due to the exploration of the importance of each template. Figure 9 shows the learned weights on Danish dataset. Several templates are highly weighted, while most of the rest approach to zero. This interpretability can be beneficial for the understanding of syntax of natural languages. According to the number of groups we used in this task, we can conclude that our MTL framework can handle hundreds of templates over a large number of instances and high dimensional data. This mainly attributes to the proposed MKL method for training in the primal by cutting plane algorithm.

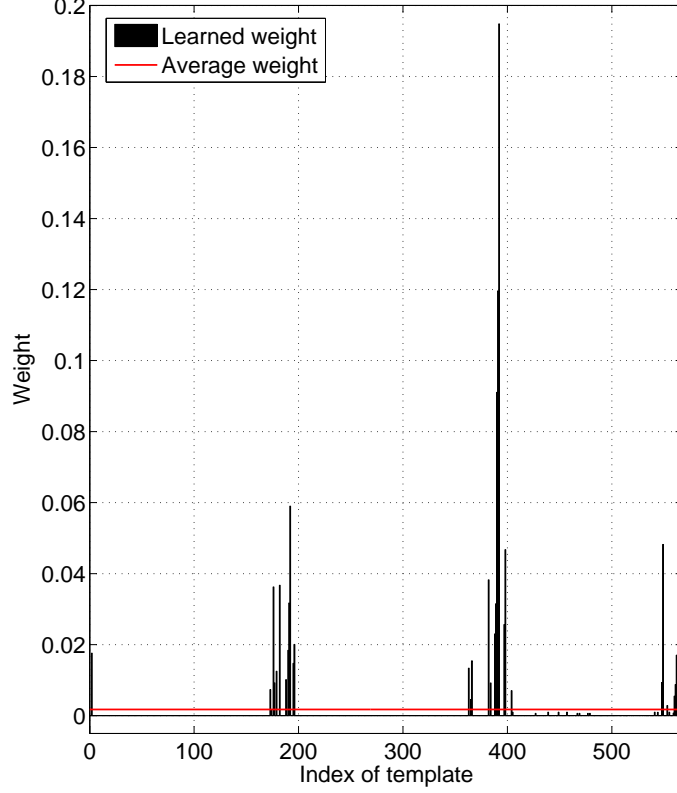


FIGURE 9. The weight of each template learned by MTLParser on Danish dataset, and the red solid line for the average weight.

6. CONCLUSION

Structured prediction is an important modeling strategy for various real world applications. Instead of modeling the structure of specific application, in this paper, we explore the underlying feature structure over the input-output pairs which contribute to the success of discriminative models, such as CRFs and Structural SVMs. Structured prediction algorithms take more effort on how to model the interdependence among the output variables, but less consideration is taken on the feature engineering which is a non-trivial task for general users. To alleviate this issue, we propose a Multiple Template Learning (MTL) paradigm to learn both the weight of each template and the structured prediction model, simultaneously. Given a set of predefined templates, the features extracted from each individual template can be naturally formed as a group. Learning the weights of these groups is formulated as a Multiple Kernel Learning (MKL) problem. We proposed to solve this MKL problem in the primal by an efficient cutting plane algorithm. MTL framework can be easily instantiated for specific applications which inherits from Structural SVMs.

Two special cases are explored in our proposed MTL framework, i.e. sequence labeling and dependency parsing. Extensive experimental results demonstrate that learning structured prediction model with weighting template can automatically interpret the importance of each templates, so users can define templates without cautions. It is helpful to prevent degradation of prediction model after poor or even conflicting templates are added by users. On the other hand, it can boost the prediction performance by weighting the features among different groups.

APPENDIX A. THE CONIC DUAL OF PROBLEM (9)

Without loss of generality, we denote $\mathbf{w} = [\mathbf{w}_1; \dots; \mathbf{w}_m]$ and $\mathbf{p}^r = [\mathbf{p}_1^r; \dots; \mathbf{p}_m^r]$. Problem (9) becomes

$$\begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2} \left(\sum_{j=1}^m \|\mathbf{w}_j\| \right)^2 + C\xi \\ \text{s.t.} \quad & \xi \geq q^r + \mathbf{w}^T \mathbf{p}^r, \forall r = 1, \dots, s \end{aligned}$$

By introducing a new variable $u \in \mathbb{R}$ and moving out summation operator from objective to be a constraint, we can obtain the equivalent optimization problem as

$$\begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2} u^2 + C\xi \\ \text{s.t.} \quad & \xi \geq q^r + \mathbf{w}^T \mathbf{p}^r, \forall r = 1, \dots, s \\ & \sum_{j=1}^m \|\mathbf{w}_j\| \leq u. \end{aligned}$$

We can further simplify above problem by introducing another variables $\rho \in \mathbb{R}^m$ such that $\|\mathbf{w}_j\| \leq \rho_j, \forall j = 1, \dots, m+1$ to be

$$\begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2} u^2 + C\xi \\ \text{s.t.} \quad & \xi \geq q^r + \mathbf{w}^T \mathbf{p}^r, \forall r = 1, \dots, s \\ & \sum_{j=1}^m \rho_j \leq u, \|\mathbf{w}_j\| \leq \rho_j, \forall j = 1, \dots, m. \end{aligned}$$

We know that $\|\mathbf{w}_j\| \leq \rho_j$ is a second-order cone constraint. Following the recipe of [5], the self-dual cone $\|\mathbf{v}_j\|_2 \leq \eta_j, \forall j = 1, \dots, m$ can be introduced to form the Lagrangian function

$$\begin{aligned} \mathcal{L}(\mathbf{w}, \xi, u, \rho; \alpha, \tau, \gamma, \mathbf{v}, \eta) = & \frac{1}{2} u^2 + C\xi - \sum_{r=1}^s \alpha_r (\xi - q^r - \mathbf{w}^T \mathbf{p}^r) - \tau \xi + \gamma \left(\sum_{j=1}^m \rho_j - u \right) \\ & - \sum_{j=1}^m (\langle \mathbf{v}_j, \mathbf{w}_j \rangle + \eta_j \rho_j), \end{aligned}$$

with dual variables $\alpha_r \in \mathbb{R}_+, \tau \in \mathbb{R}_+, \gamma \in \mathbb{R}_+$. The derivatives of the Lagrangian with respect to the primal variables have to vanish which leads to the following

KKT conditions:

$$(28) \quad \mathbf{v}_j = \sum_{r=1}^s \alpha_r \mathbf{p}_j^r, \forall j = 1, \dots, m$$

$$(29) \quad C - \sum_{r=1}^s \alpha_r - \tau = 0$$

$$(30) \quad u = \gamma$$

$$(31) \quad \gamma = \rho_j, \forall j = 1, \dots, m$$

By substituting all the primal variables with dual variables by above KKT conditions, we can obtain the following dual problem,

$$\begin{aligned} \max_{\alpha, \gamma} \quad & -\frac{1}{2}\gamma^2 + \sum_{r=1}^s \alpha_r q^r \\ \text{s.t.} \quad & \left\| \sum_{r=1}^s \alpha_r \mathbf{p}_j^r \right\| \leq \gamma, \forall j = 1, \dots, m \\ & \sum_{r=1}^s \alpha_r \leq C, \alpha_r \geq 0, \forall r = 1, \dots, s \end{aligned}$$

By setting $\theta = \frac{1}{2}\gamma^2$ and $\mathcal{A}_s = \{\sum_{r=1}^s \alpha_r \leq C, \alpha_r \geq 0, \forall r = 1, \dots, s\}$, we can reformulate above problem as

$$\begin{aligned} \max_{\theta, \alpha \in \mathcal{A}_s} \quad & -\theta + \sum_{r=1}^s \alpha_r q^r \\ \text{s.t.} \quad & \frac{1}{2}\alpha^T Q^j \alpha \leq \theta, \forall j = 1, \dots, m \end{aligned}$$

where $Q_{r,r'}^j = \langle \mathbf{p}_j^r, \mathbf{p}_j^{r'} \rangle$. According to the property of self-dual cone, we can obtain the primal solution from its dual as $\mathbf{w}_j = \mu_j \mathbf{v}_j = \mu_j \sum_{r=1}^s \alpha_r \mathbf{p}_j^r$ where μ_j is the dual variable of the j^{th} quadratic constraint such that $\sum_{j=1}^m \mu_j = 1, \mu_j \in \mathbb{R}_+, \forall j = 1, \dots, m$.

REFERENCES

- [1] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *International Conference on Machine Learning*, 2003.
- [2] E. D. Andersen and A. D. Andersen. *The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm*. Kluwer Academic Publishers, 2000.
- [3] G. Attardi. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of CoNLL*, 2006.
- [4] F. R. Bach, R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *the 21st International Conference on Machine Learning*, 2004.
- [5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK., 2004.
- [6] Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.
- [7] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, 2004.
- [8] J. Eisner. Three new probabilistic models for dependency parsing: an exploration. In *International Conference on Computational Linguistics (COLING)*, 1996.
- [9] T. Emerson. The second interational chinese word segmentation bakeoff. In *the Fourth SIGHAN workshop on Chinese Language Processing*, 2005.

- [10] Y. Feng, R. Huang, and L. Sun. Two step chinese named entity recognition based on conditional random fields models. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*, 2008.
- [11] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.
- [12] T. Joachims, T. Finley, and Y. Chun-Nam. Cutting-plane training of structural SVMs. *Machine Learning Journal*, 77(1):27–59, 2009.
- [13] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.
- [14] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
- [15] A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. Xing. Online multiple kernel learning for structured prediction. *Arxiv preprint arXiv:1010.2770*, 2010.
- [16] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of The Seventh Conference on Natural Language Learning (CoNLL-2003)*, 2003.
- [17] R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *the 43rd Annual Meeting of the ACL*, 2005.
- [18] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithm. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, 2005.
- [19] T. Nakagawa. Multilingual dependency parsing using global features. In *Proceeding of EMNLP-CoNLL*, 2007.
- [20] J. Nivre, J. Hall, and J. Nilsson. Memory-based dependency parsing. In *Proceedings of CoNLL*, 2004.
- [21] J. Nivre and R. McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceeding of ACL-08: HLT*, 2008.
- [22] D. Okanohara, Y. Miyao, Y. Tsuruoka, and J. Tsujii. Improving the scalability of semi-markov conditional random fields for named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006.
- [23] F. Peng, F. Feng, and A. McCallum. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics (COLING '04)*, 2004.
- [24] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.
- [25] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.
- [26] A. Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution*. PhD thesis, University of Pennsylvania, 1998.
- [27] S. Sonnenburg, G. Rätsch, B. Schölkopf, and G. Rätsch. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 2006.
- [28] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Neural Information Processing System*, 2003.
- [29] R. Tomioka and T. Suzuki. SpicyMKL. *Arxiv preprint arXiv:0909.5026*, 2009.
- [30] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.
- [31] D. L. Vail, J. Lafferty, and M. M. Veloso. Feature selection in conditional random fields for activity recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.
- [32] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transaction on Information Theory*, pages 260–269, 1967.
- [33] M. Wang, N. A. Smith, and T. Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of EMNLP-CoNLL*, 2007.

- [34] Z. Xu, R. Jin, I. King, and M. Lyu. An extended level method for efficient multiple kernel learning. In *Annual Conference on Neural Information Processing Systems*, 2009.
- [35] N. Xue. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8:29–48, 2003.
- [36] H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, 2003.
- [37] L. Yao, S. Riedel, and A. McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010.
- [38] C. Zhang, Z. Chen, and G. Hu. A chinese word segmentation system based on structured support vector machine utilization of unlabeled text corpus. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*, 2010.
- [39] A. Zien and C. Ong. Multiclass multiple kernel learning. In *Proceedings of the 21th International Conference on Machine Learning*, 2007.